CREATION OF BURP EXTENSIONS

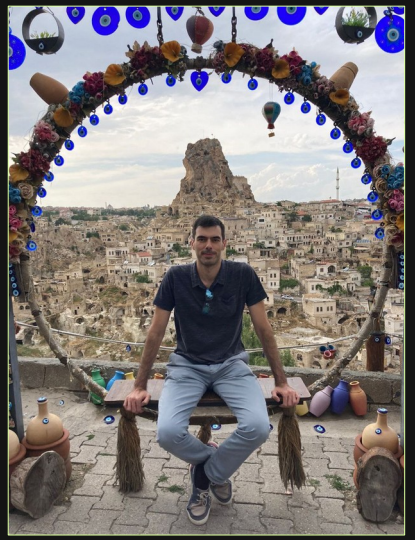
Antonio Lara
(dh0ck)

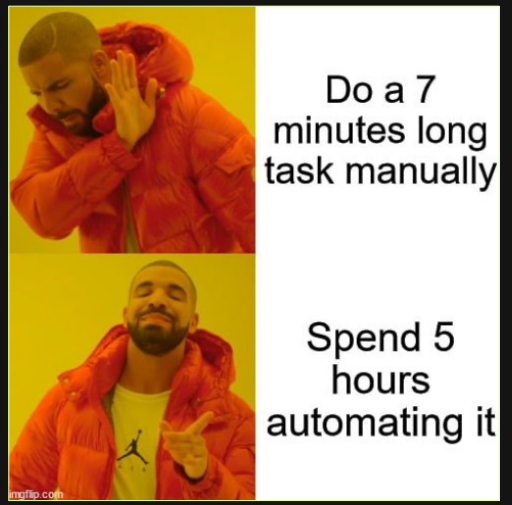
ABOUT ME

- Physics background, later transitioned to IT
- Currently work as a Security Engineer at TWS
- I like to automate stuff
- Author of Burp Extension:

  Header Issue Reporter

https://github.com/dh0ck
https://www.linkedin.com/in/dh0ck/





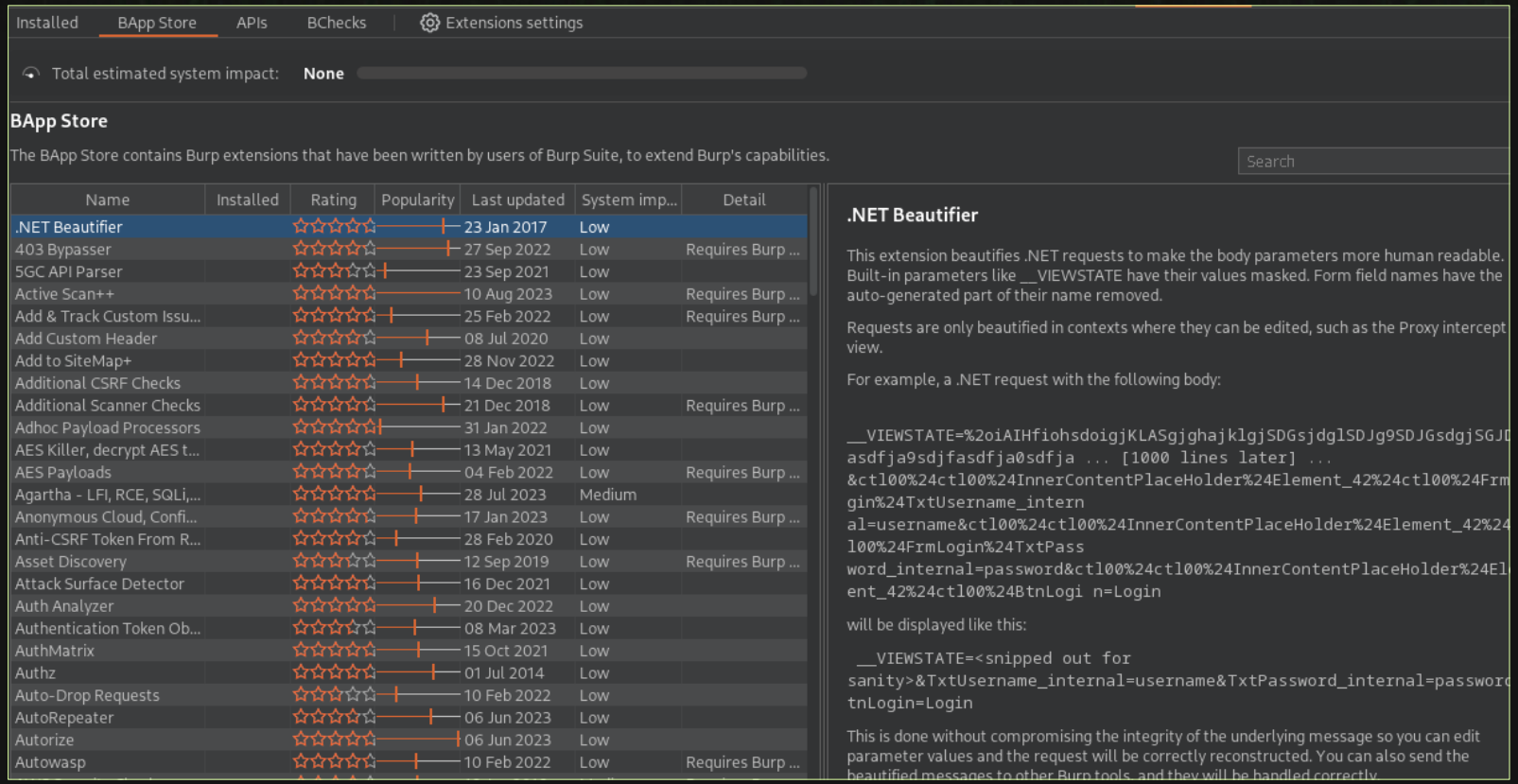
BURP EXTENDER: INTRODUCTION

- Burp has an API to interact with some of its tools or allow us to integrate them in our workflow
- Not everything is accessible via API in Burp

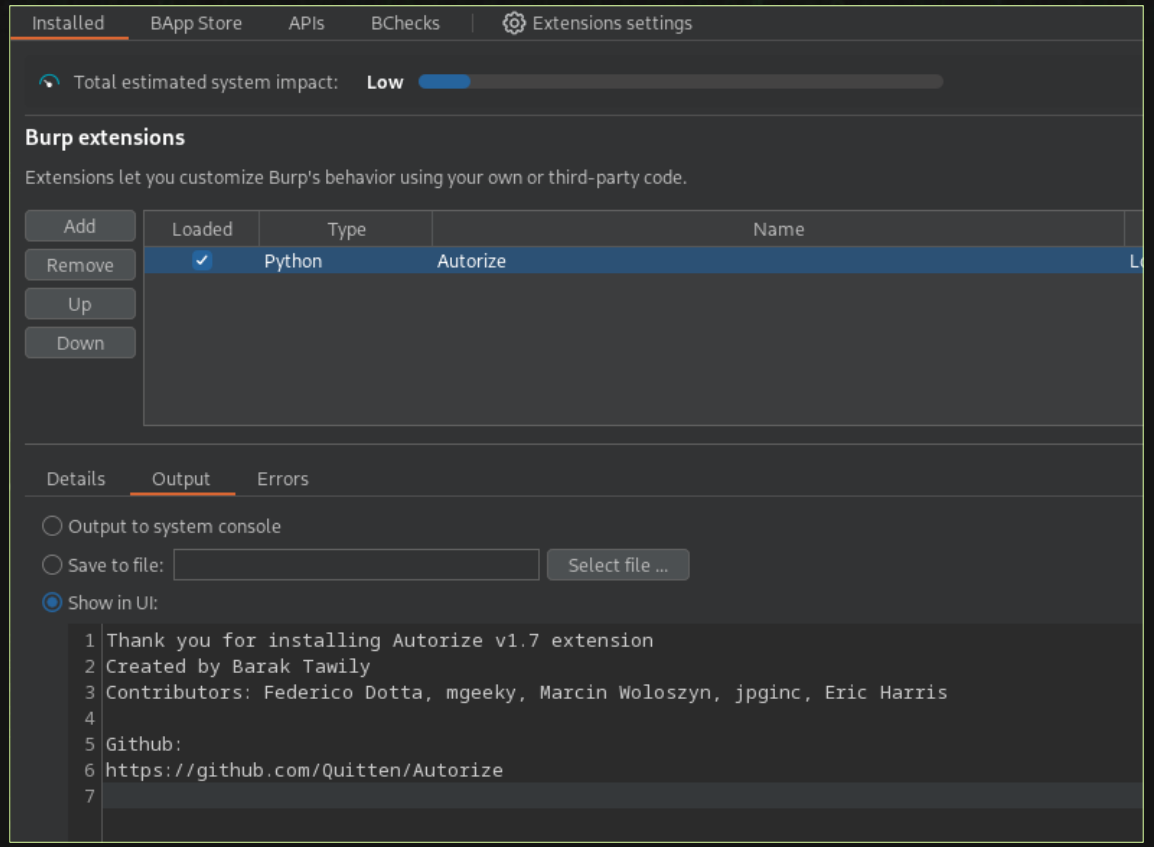
OUTLINE

- Intro to Burp Extender
- OOP
- Burp API
- Creating basic extensions
- Submitting an extension to the BApp store

BAPP STORE


Installed
BApp Store
APIs
BChecks
Extensions settings
Total estimated system impact:
None
BApp Store
The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.
Search
Name
Installed
Rating
Popularity
Last updated
System imp...
Detail
.NET Beautifier
23 Jan 2017
Low
403 Bypasser
27 Sep 2022
Low
Requires Burp ...
5GC API Parser
23 Sep 2021
Low
Active Scan++
10 Aug 2023
Low
Requires Burp ...
Add & Track Custom Issu...
25 Feb 2022
Low
Requires Burp ...
Add Custom Header
08 Jul 2020
Low
Add to SiteMap+
28 Nov 2022
Low
Additional CSRF Checks
14 Dec 2018
Low
Additional Scanner Checks
21 Dec 2018
Low
Requires Burp ...
Adhoc Payload Processors
31 Jan 2022
Low
AES Killer, decrypt AES t...
13 May 2021
Low
AES Payloads
04 Feb 2022
Low
Requires Burp ...
Agartha - LFI, RCE, SQLi,...
28 Jul 2023
Medium
Anonymous Cloud, Confi...
17 Jan 2023
Low
Requires Burp ...
Anti-CSRF Token From R...
28 Feb 2020
Low
Asset Discovery
12 Sep 2019
Low
Requires Burp ...
Attack Surface Detector
16 Dec 2021
Low
Auth Analyzer
20 Dec 2022
Low
Authentication Token Ob...
08 Mar 2023
Low
AuthMatrix
15 Oct 2021
Low
Authz
01 Jul 2014
Low
Auto-Drop Requests
10 Feb 2022
Low
AutoRepeater
06 Jun 2023
Low
Autorize
06 Jun 2023
Low
Autowasp
10 Feb 2022
Low
Requires Burp ...
.NET Beautifier
This extension beautifies .NET requests to make the body parameters more human readable. Built-in parameters like __VIEWSTATE have their values masked. Form field names have the auto-generated part of their name removed.
Requests are only beautified in contexts where they can be edited, such as the Proxy intercept view.
For example, a .NET request with the following body:
__VIEWSTATE=%2oiAIHfiohsdoigjKLASgjghajklgjSDGsjdglSDJg9SDJGsdgjSGJ
asdfja9sdjfasdfja0sdfja ... [1000 lines later] ...
&ctl00%24ctl00%24InnerContentPlaceHolder%24Element_42%24ctl00%24Frm
gin%24TxtUsername_intern
al=username&ctl00%24ctl00%24InnerContentPlaceHolder%24Element_42%24
l00%24FrmLogin%24TxtPass
word_internal=password&ctl00%24ctl00%24InnerContentPlaceHolder%24El
ent_42%24ctl00%24BtnLogi n=Login
will be displayed like this:
__VIEWSTATE=<snipped out for
sanity>&TxtUsername_internal=username&TxtPassword_internal=passwor
tnLogin=Login
This is done without compromising the integrity of the underlying message so you can edit parameter values and the request will be correctly reconstructed. You can also send the beautified messages to other Burp tools, and they will be handled correctly.

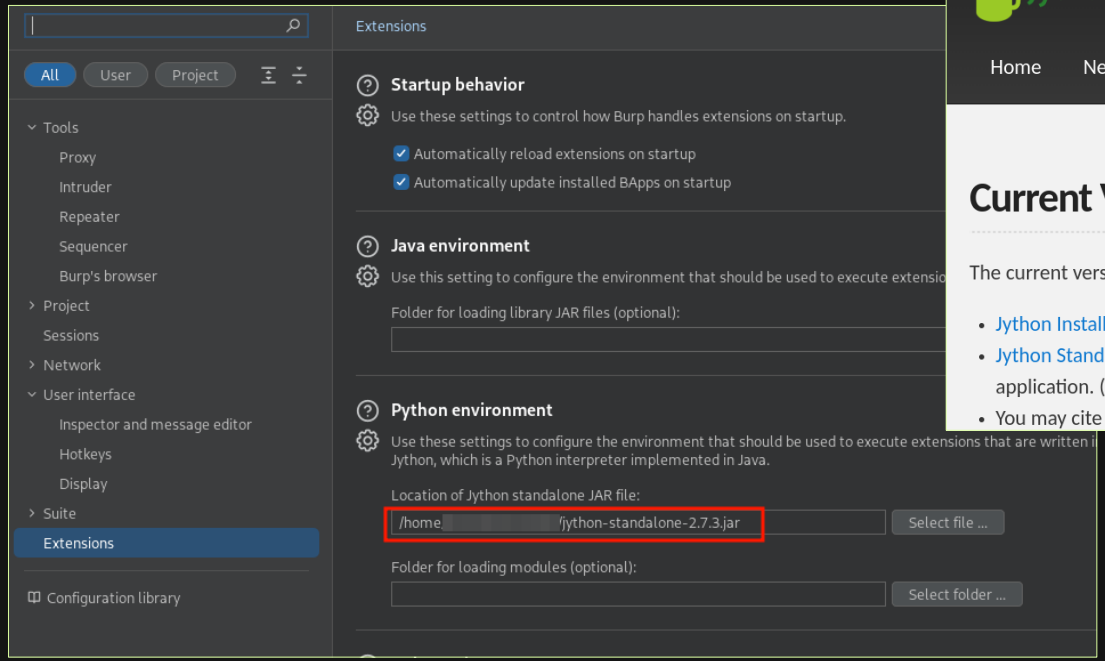
MAIN EXTENDER WINDOW

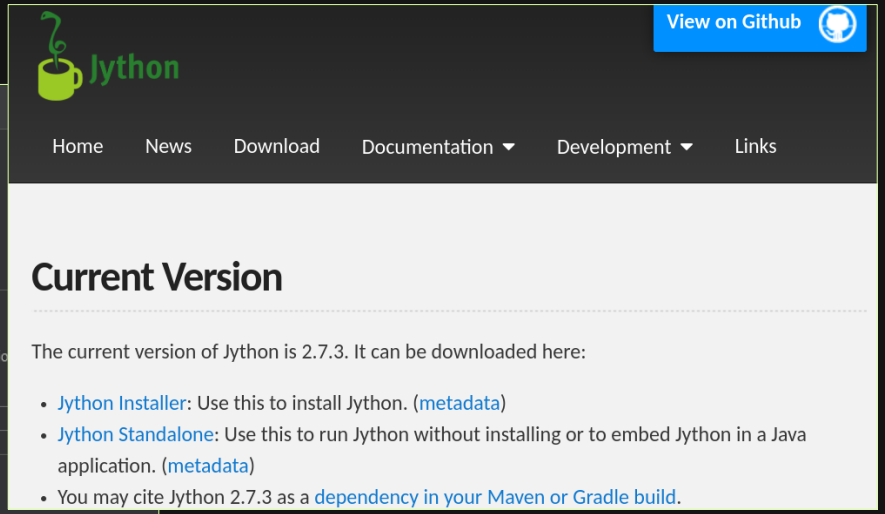


RUNNING PYTHON EXTENSIONS

https://repo1.maven.org/maven2/org/python/jython-standalone/2.7.3/jython-standalone-2.7.3.jar

https://www.jython.org/download.html





OBJECT ORIENTED PROGRAMMING

OBJECT ORIENTED PROGRAMMING

- "Style" of programming that uses "objects" (logical entities with properties (features) and methods (functionality)).
- Usual concepts in OOP:
  - Class
  - Instance/Object
  - Properties
  - Methods
  - Inheritance
  - Polimorphism
  - Encapsulation

Not necessary to know this, but it helps to understand how to work with extensions

CLASS VS INSTANCE

Class → Blueprint
(doesn't exist "in the real world")

AtAt.py

```
class ATAT:
    head = True

    def __init__(self):
        self.legs = 4
        self.armour = True
```

Create instance

```
atat1 = AtAt.ATAT()
```

Instance → Object
(created from a class instructions)

"self" is a way for objects to refer to themselves. Other languages use "this"

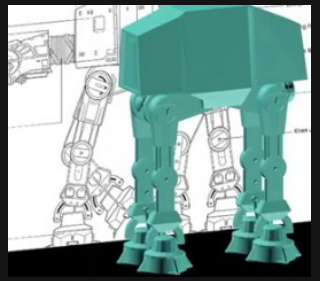
PROPERTIES

Can be accessed with the “.” operator
New properties can be added to the instance
Existing properties can be updated

```
>>> atat1.color = "pink"
>>> atat1.color
'pink'
```

```
>>> atat1.material = "wood"
>>> atat1.material
'wood'
```

```
>>> atat1.head = False
>>> atat1.head
False
```



```
>>> atat1.dead = True
>>> atat1.dead
True
```

THE CLASS DOESN'T CHANGE, ONLY THE INSTANCE

Instance →

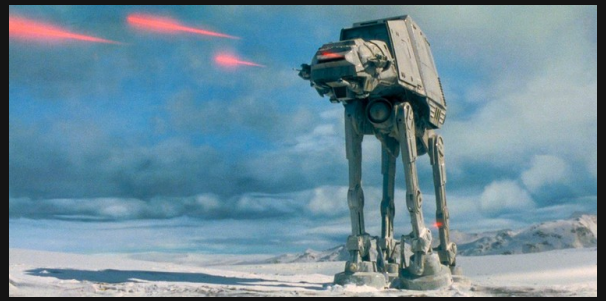
Class →

```
>>> dir(atat1)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__form
at__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_s
ubclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclas
shook__', '__weakref__', 'armour', 'color', 'dead', 'head', 'legs', 'material',
'status']
>>> dir(AtAt.ATAT)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__form
at__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_s
ubclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclas
shook__', '__weakref__', 'head']
```

METHODS

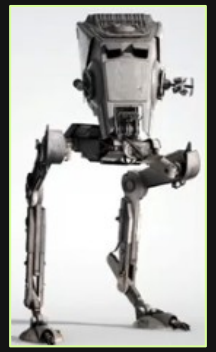
```
class ATAT:

    head = True
    position = 0

    def __init__(self):
        self.legs = 4
        self.armour = True
        self.battery = 100

    def walk(self, distance):
        self.position += distance
        self.battery -= distance * 0.1

    def fire(self, seconds):
        self.battery -= seconds * 0.01
        return self.battery
```

Class methods (present in all instances)



INHERITANCE

```
class ATXX:
    head = True
    position = 0

    def __init__(self):
        self.battery = 100

    def wait(self, distance):
        self.position += distance
        self.battery -= distance * 0.1
        return self.position

class ATAT(ATXX):
    def __init__(self):
        super().__init__()
        self.legs = 4
        self.armour = 50

class ATST(ATXX):
    def __init__(self):
        super().__init__()
        self.armour = 10
        self.legs = 2

class ATPT(ATXX):
    def __init__(self):
        super().__init__()
        self.legs = 2
```



INHERITANCE

Inheritance in programming courses

```java
package practice;

class Cat extends Animal {
    private String name;

    Cat(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Easy! A cat is an instance of an animal!

Inheritance in reality

```java
package com.company.big;

import java.util.*;

class DefaultFeedListener extends ConstraintEventConverter {
    private StreamListener streamListener;
    private RetroEncabulator retroEncabulator;
    private StringExtender stringExtender;
    private WebCompatAction webCompatAction;

    DefaultFeedListener(
        StreamListener streamListener,
        RetroEncabulator retroEncabulator,
        StringExtender stringExtender,
        WebCompatAction webCompatAction
    ) {
        super();
        this.streamListener = streamListener;
        this.retroEncabulator = retroEncabulator;
        this.stringExtender = stringExtender;
        this.webCompatAction = webCompatAction;
    }

    public void onRequestCompatibleActivities() {
        streamListener.createNewEventListener();
        resetCompatibleVariables();
    }

    public void EnableConditionalCallback(String arg) throws ArgumentException {
        try {
            if (arg.equals("typeset:00")) {
                modifyInitialCall();
            }
        } catch (ArgumentException e) {
            throw e;
        }
    }
```

???

ABSTRACT CLASSES

- Burp APIs are Java interfaces
- Interface vs abstract class in Java
- Abstract class: at least an abstract (empty) method. The implementation happens in children

```
public class Shape {
    private int sides;
    public Shape() {
        this.sides = 0;
    }

    public float area(){
        return 0.0f;
    }
}
```

```
public class Circle extends Shape {
    public float radius;

    public float area() {
        return 3.14f * this.radius *
        this.radius;
    }
}
```

IMPLEMENTING, OVERRIDING

```java
public class Shape {
    private int sides;
    public Shape() {
        this.sides = 0;
    }

    public float area(){
        return 0.0f;
    }
}
```

```java
public class Circle extends Shape {
    public float radius;

    public float area() {
        return 3.14f * this.radius * this.radius;
    }
}
```

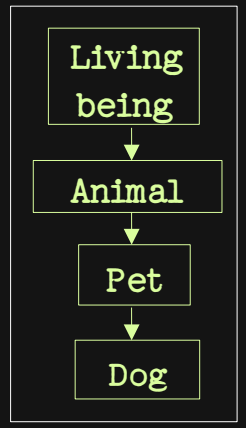
```java
public abstract class Shape {
    private int sides;
    public Shape() {
        this.sides = 0;
    }

    public abstract float area() {
    }
}
```

```java
public class Circle extends Shape {
    public float radius;

    @Override
    public float area() {
        return 3.14f * this.radius * this.radius;
    }
}
```

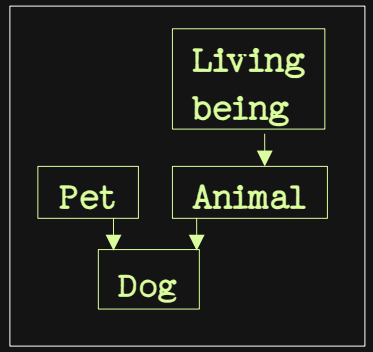
INTERFACES

Java has no multiple inheritance ⟸ Solved with interfaces (interfaces are implemented, not extended)

YES



NO




```java
public interface DogLike {
    void bark();
    void eat();
}
```

Methods in interfaces have no logic. Cannot be instantiated, only implemented

```java
public class Dog extends Pet implements DogLike {
    @Override
    public void bark(){
        // do something
    }

    @Override
    public void eat(){
        // eat something
    }
}
```

- A child class can only extend an abstract class, but can implement any number of interfaces
- Abstract class: at least a method is abstract and there can be non abstract methods (implemented)
- Interface: all methods are abstract, non has been coded, they are only “declared”

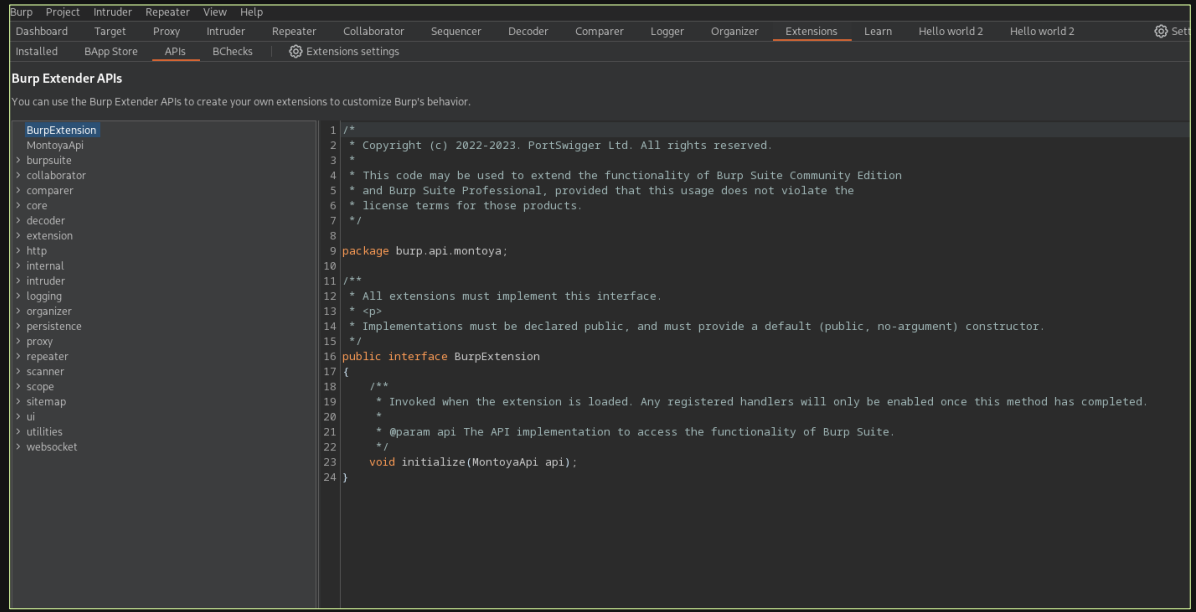
BURP API

BURP API SOURCE CODE

Montoya API (current) → https://github.com/PortSwigger/burp-extensions-montoya-api

- Examples → https://github.com/PortSwigger/burp-extensions-montoya-api-examples

Also available in API tab



- Wiener API (legacy) → https://github.com/PortSwigger/burp-extender-api

OLD VS NEW BURP API

- Previously: Wiener API (https://github.com/PortSwigger/burp-extender-api)
- Currently: Montoya API (why, Portswigger???)

https://portswigger.net/blog/new-burp-suite-api-we-want-your-feedback

If you follow the Burp Suite roadmap, then you'll know that we're working on a complete rewrite of the "Wiener" API used in Burp Suite Professional and Burp Suite Community Edition. The new API is codenamed "Montoya", and will eventually expose much more of Burp Suite's core functionality - providing richer capabilities for writers of Burp extensions.

As one of the devs working on this project, I'm pleased to announce that things are going well. The feature-parity Montoya API has been available since Burp Suite 2022.9.5, and we're going to be adding more functionality very soon (like in 2022.11, which added API support for WebSocket listeners).

EXTENSION EXAMPLES

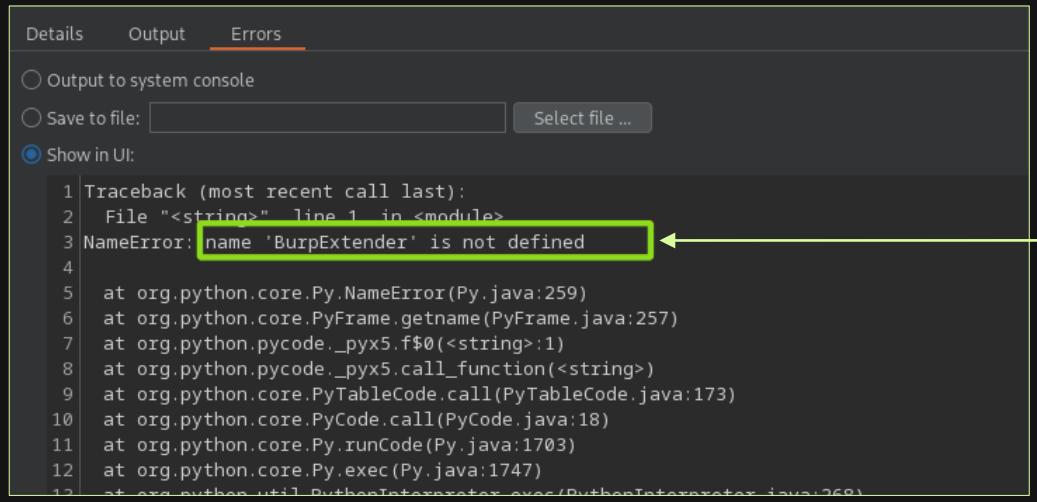
SIMPLE EXTENSION

```
class BurpExtender(IBurpExtender):
    # Function indicated in IBurpExtender interface
    def registerExtenderCallbacks(self, callbacks):
        callbacks.setExtensionName("hello woorld")
        callbacks.printOutput("Hello!")
        return
```

Inherits from the IBurpExtender class

Necessary to register Extender callbacks

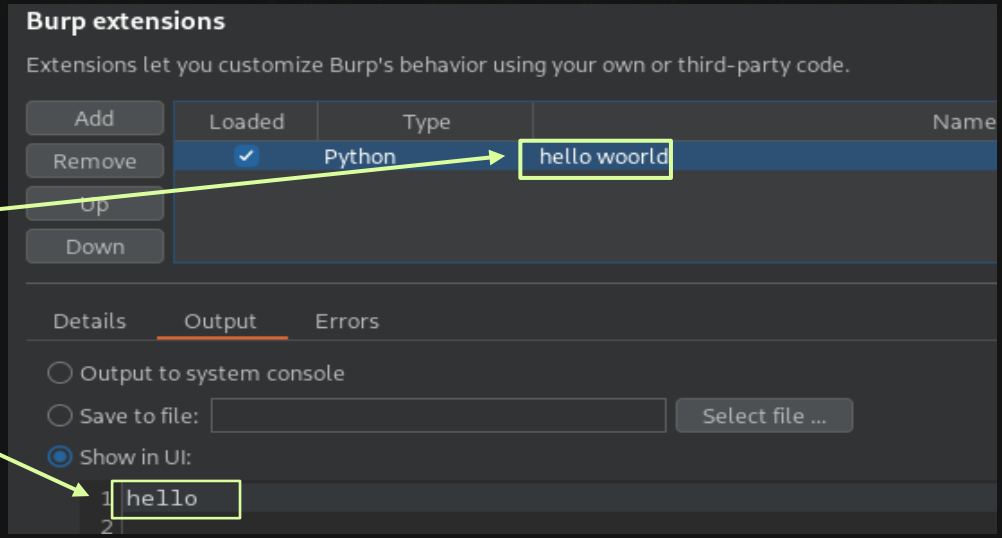
Name of the extension



Necessary to have class called “BurpExtender”

EXTENSION NAME, OUTPUT

```
class BurpExtender(IBurpExtender):
    # Function indicated in IBurpExtender interface
    def registerExtenderCallbacks(self, callbacks):
        callbacks.setExtensionName("hello woorld")
        callbacks.printOutput("hello!")
        return
```



Method Summary

All Methods | Instance Methods | Abstract Methods | Deprecated Methods

Modifier and Type	Method and Description
void	**addScanIssue**(IScanIssue issue) This method is used to register a new Scanner issue.
void	**addSuiteTab**(ITab tab) This method is used to add a custom tab to the main Burp Suite window.
void	**addToSiteMap**(IHttpRequestResponse item) This method can be used to add an item to Burp's site map with the specified request/response details.
IHttpRequestResponseWithMarkers	**applyMarkers**(IHttpRequestResponse httpRequestResponse, java.util.List<int[]> requestMarkers, java.util.List<int[]> responseMarkers) This method is used to apply markers to an HTTP request or response, at offsets into the message that are relevant for some particular purpose.

https://portswigger.net/burp/extender/api/burp/iburpextendercallbacks.html

IBURPEXTENDER

demo2.py

```python
class BurpExtender(IBurpExtender):

    def registerExtenderCallbacks(self, callbacks):
        # set our extension name
        callbacks.setExtensionName("Hello world extension")

        # obtain our output and error streams
        stdout = PrintWriter(callbacks.getStdout(), True)
        stderr = PrintWriter(callbacks.getStderr(), True)

        # write a message to our output stream
        stdout.println("Hello output")

        # write a message to our error stream
        stderr.println("Hello errors")

        # write a message to the Burp alerts tab
        callbacks.issueAlert("Hello alerts")

        # throw an exception that will appear in our error
        stream
        raise RuntimeException("Hello exception")
```

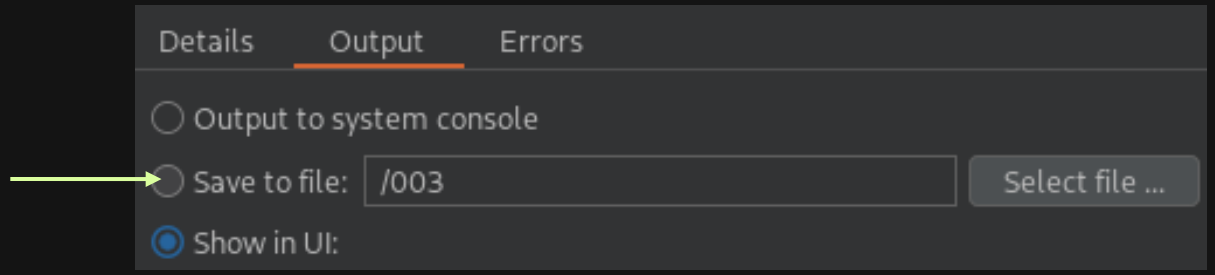
IBurpExtender.java

```java
package burp;

/*
...
*/
public interface IBurpExtender
{
    /**
    ...
    */
    void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks);
}
```

◀ https://github.com/PortSwigger/example-hello-world/blob/master/python/HelloWorld.py

SAVING LONG OUTPUT

- The “Show in UI” window can only hold ~100 lines. Good for debugging, but it’s safer to save to an external file.
- But output to an external file can always be saved programmatically from inside the extension

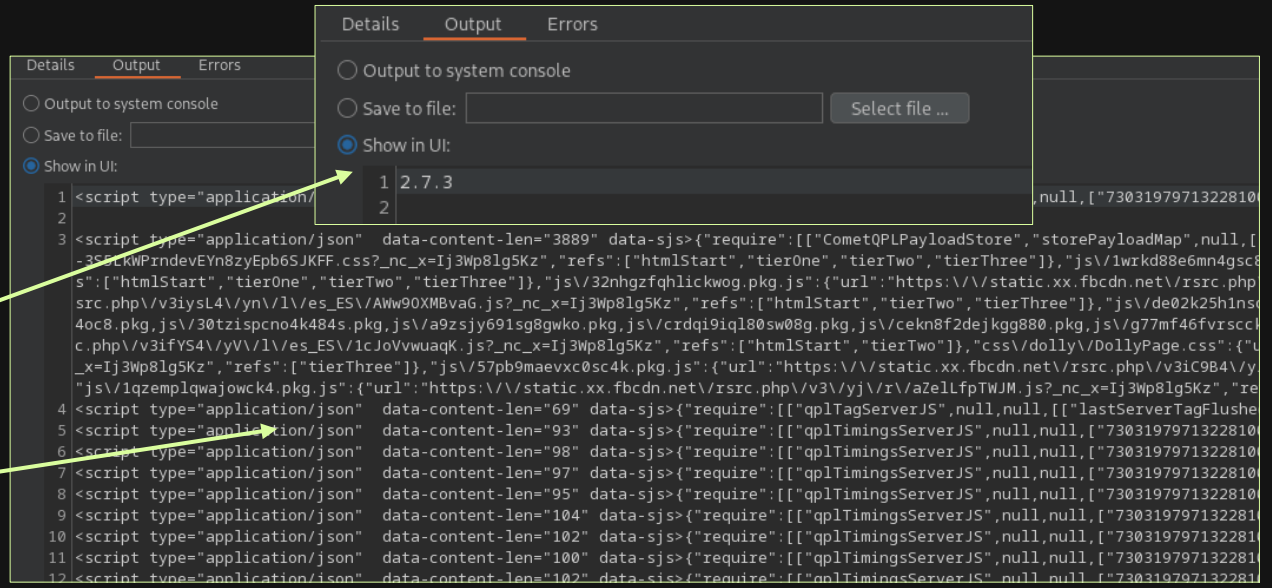


PIP FOR JYTHON

```
java -jar jython-standalone-2.7.3.jar -m ensurepip
java -jar jython-standalone-2.7.3.jar -m pip install requests
```

Necessary to be able to install pip packages

```
from burp import IBurpExtender

class BurpExtender(IBurpExtender):

    def registerExtenderCallbacks(self, callbacks):
        callbacks.setExtensionName("Requests")
        import platform
        callbacks.printOutput(platform.python_version()
        )

        import requests
        a = requests.get("http://meta.com")
        callbacks.printOutput(a.text)
        return
```



DEMO 4: CONTEXT MENU

Demo4.py

```python
class BurpExtender(IBurpExtender, IContextMenuFactory):
    def registerExtenderCallbacks(self, callbacks):
        self._callbacks = callbacks
        callbacks.setExtensionName("New menus")
        callbacks.registerContextMenuFactory(self)
        return

    def createMenuItems(self, context_menu):
        self.context = context_menu
        submenu = JMenu("Submenu")
        menu_list = ArrayList()
        menu_list.add(JMenuItem("Item 1",
        actionPerformed=self.new_menu_1))
        menu_list.add(submenu)
        submenu.add(JMenuItem("Item 2", actionPerformed=self.new_menu_2))
        return menu_list

    def new_menu_1(self, event):
        pass

    def new_menu_2(self, event):
        pass
```

IContextMenuFactory.java

```java
package burp;

/*
...
*/

import javax.swing.JMenuItem;
import java.util.List;

/**
* Extensions can implement this interface and then call
...
*/
public interface IContextMenuFactory
{
    /**
    * This method will be called by Burp when the user invokes a context
    menu
    ...
    */
    List<JMenuItem> createMenuItems(IContextMenuInvocation invocation);
}
```

CALLBACK HELPERS

```
IRequestinfo analyzeRequest(IHttpRequestResponse request);
IRequestinfo analyzeRequest(IHttpService httpService, byte[] request);
IRequestinfo analyzeRequest(byte[] request);
IResponseinfo analyzeResponse(byte[] response);
IParameter getRequestParameter(byte[] request, String parameterName);
String urlDecode(String data);
String urlEncode(String data);
byte[] urlDecode(byte[] data);
byte[] urlEncode(byte[] data);
byte[] base64Decode(String data);
byte[] base64Decode(byte[] data);
String base64Encode(String data);
String base64Encode(byte[] data);
byte[] stringToBytes(String data);
String bytesToString(byte[] data);
int indexOf(byte[] data, byte[] pattern, boolean caseSensitive, int from, int to);
byte[] buildHttpMessage(List<String> headers, byte[] body);
byte[] buildHttpRequest(URL url);
byte[] addParameter(byte[] request, IParameter parameter);
byte[] removeParameter(byte[] request, IParameter parameter);
byte[] updateParameter(byte[] request, IParameter parameter);
byte[] toggleRequestMethod(byte[] request);
IHttpService buildHttpService(String host, int port, String protocol);
IHttpService buildHttpService(String host, int port, boolean useHttps);
IParameter buildParameter(String name, String value, byte type);
IScannerInsertionPoint makeScannerInsertionPoint(String insertionPointName, byte[] baseRequest, int from, int to);
IResponseVariations analyzeResponseVariations(byte[]... responses);
IResponseKeywords analyzeResponseKeywords(List<String>, keywords, byte[]... responses);
```

callback_helpers.py

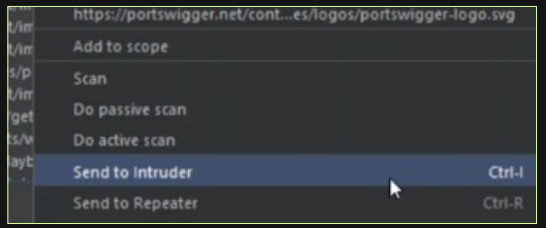
Helpers are useful methods that Burp makes available, and that can facilitate the processing of data

IBurpExtenderCallbacks has the getHelpers() method, Which return an object of type IextensionHelpers. Such Objects have methods that help process requests and reponses

Color legend:
- Data type returned by function
- xx[] = array of objects of type xx

CONTEXT MENU INVOCATION

```
static final byte CONTEXT MESSAGE_EDITOR_REQUEST = 0;
static final byte CONTEXT MESSAGE_EDITOR_RESPONSE = 1;
static final byte CONTEXT MESSAGE_VIEWER_REQUEST = 2;
Static final byte CONTEXT MESSAGE_VIEWER_RESPONSE = 3;
static final byte CONTEXT TARGET_SITE_MAP TREE = 4;
static final byte CONTEXT_TARGET_SITE_MAP TABLE = 5;
static final byte CONTEXT_PROXY_HISTORY = 6;
static final byte CONTEXT SCANNER_RESULTS = 7;
static final byte CONTEXT INTRUDER_PAYLOAD_POSITIONS = 8;
static final byte CONTEXT_INTRUDER_ATTACK_RESULTS = 9;
static final byte CONTEXT SEARCH_RESULTS = 10;
InputEyent getInputEvent();
int getToolFlag();
byte getInvocationContext();
int[] getSelectionBounds();
IHttpRequestResponse[] getSelectedMessages();
IScanIssue[] getSelectedIssues();
```



Example of context menu:
Send to Intruder

Contents of IContextMenuInvocation.java

⬅ Access selected messages in a table

OBJECT FOR MESSAGES

- IHttpRequestResponse is a Data structure that contains:
  - Request
  - Response
  - Comment
  - Highlight
  - Etc

getProxyHistory

`IHttpRequestResponse[] getProxyHistory()`

This method returns details of all items in the Proxy history.

Returns:

The contents of the Proxy history.

- We can get its value, or change it dynamically
- getProxyHistory returns an array of object of type IHttpRequestResponse

  - `self._callbacks.getProxyHistory()`

INFORMATION FROM REQUEST/RESPONSE

```
byte[] getRequest();
void setRequest(byte[] message);
byte[] getResponse();
void setResponse(byte[] message);
String getComment();
void setComment(String comment);
String getHighlight();
void setHighlight(String color);
IHttpService getHttpService();
void setHttpService(IHttpService httpService);
```

```
byte[] stringToBytes(String data);
String bytesToString(byte[] data);
```

Demo5.py

```
public interface
IHttpService
String getHost();
int getPort();
String getProtocol();
```

EXTENSION WITH TAB

Demo_tab1.py

```python
from burp import IBurpExtender, ITab
from javax.swing import JPanel
from java.awt import BorderLayout

class BurpExtender(IBurpExtender, ITab):
    def registerExtenderCallbacks(self, callbacks):
        callbacks.setExtensionName("hello woorld")
        callbacks.printOutput("hello")
        callbacks.addSuiteTab(self)
        return

    # tab name
    def getTabCaption(self):
        return "Hello world"

    # GUI details
    def getUiComponent(self):
        panel = JPanel(BorderLayout())
        return panel
```

ITab.java

```java
package burp;
import java.awt.Component;
/**
...
*/
public interface ITab
{
    /**
    ...
    */
    String getTabCaption();
    /**
    ...
    */
    Component getUiComponent();
}
```

RE-ENABLE HTML RENDERING

> “HTML rendering in Swing components has been disabled as a security measure in version 2022.3 and above.”

https://forum.portswigger.net/thread/burp-suit-professional-2022-3-1-is-not-supporting-html-tag-3160df4a

Examples taken from Header Issue Reporter exension:

```
# Label HTML rendering
self.extra_info_label1.putClientProperty("html.disable", None)

# Button HTML rendering
make_curr_selection_permanent_button.putClientProperty("html.disable", None)
make_curr_selection_permanent_button.setForeground(Color.WHITE)
make_curr_selection_permanent_button.setBackground(Color(10,101,247))

# Table HTML rendering
self.model_tab_resp = IssueTableModel([["",""]], self.colNames)
self.table_tab_resp = IssueTable(self.model_tab_resp, "tab")
self.table_tab_resp.getColumnModel().getColumn(0).setCellRenderer(RawHtmlRenderer())
self.table_tab_resp.getColumnModel().getColumn(1).setCellRenderer(RawHtmlRenderer())
```

SUBMITTING AN EXTENSION TO THE BAPP STORE

BEFORE YOU SUBMIT AN EXTENSION TO THE BAPP STORE

- What value does your extension provide?
- Does it have errors?
- Is it tested? Will you maintain it?
- Does it use Burp functions instead of Python/Java/Ruby functions when necessary?

THANKS FOR YOUR ATTENTION

Questions? Reach out on LinkedIn or Telegram:


https://www.linkedin.com/in/dh0ck/


@dh0ck

Demos can be found at:
https://github.com/dh0ck/Burp_Extender_Workshop_MystikCon_2023